March 2013

Efficient Methods for Calculating Equivalent Resistance Between Nodes of a Highly Symmetric Resistor Network

Jeremy Thomas Moody
Worcester Polytechnic Institute

Efficient Methods for Calculating Equivalent Resistance Between Nodes of a Highly Symmetric Resistor Network

A Major Qualifying Project

Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

in partial fulfillment of the requirements for the

Degree of Bachelor of Science

By

Jeremy T. Moody

Date: March 29, 2013

Advisors:
Prof. Padmanabhan K. Aravind
Department of Physics

Prof. Brigitte Servatius
Department of Mathematical Sciences

Abstract

Classic Kirchhoff methods for solving resistor network problems rapidly become unwieldy as the network size grows. For all but trivial networks, the number of equations to be solved makes the task tedious and error-prone. We present an algorithm for generating and solving the Kirchhoff equations for an arbitrary N-node network, given an NxN matrix representation of the network. Using the algorithm and van Steenwijk's symmetry method, we replicated his solutions for the effective resistance between nodes of Platonic polyhedral networks. We derived new solutions for Archimedean and Catalan polyhedral networks, 4D polytopes, and N-dimensional hypercubes. Finally, we constructed physical models of several networks and measured the resistance between nodes, validating our calculated results.

Acknowledgments

Padmanabhan K. Aravind

Brigitte Servatius

For their advice and assistance throughout the project.

Stephen J. Bitar

For his help designing the printed circuit boards used to test our models.

Contents

List of Tables

List of Figures

Executive Summary

A resistor network is a collection of nodes connected by resistors. This project addresses efficient methods for calculating the equivalent resistance between nodes of highly symmetric networks. This problem has been investigated previously by van Steenwijk (1998), who studied networks of resistors that form the edges of the Platonic solids, and by Tretiak and Huang (1965), who derived the resistance between opposite nodes of an n-dimensional hypercube network. This project presents new results for several finite networks that have not been considered earlier.

We start by discussing a basic method for calculating effective resistances in resistor networks, based on circuit laws developed by Gustav Kirchhoff in 1845, and we show how to use computer algorithms to do these calculations automatically, given the Kirchhoff equations of the network. For all but trivial networks, the number of equations to be solved makes the task tedious and error-prone. For this reason, we designed and implemented an algorithm (presented in the appendix) for generating the Kirchhoff equations for an arbitrary network of N nodes, given a simple NxN matrix representation of the network.

We then discuss the "Symmetry Method" introduced by van Steenwijk for simplifying Platonic polyhedral networks. From there, we demonstrate the use of the symmetry method with networks of semi-regular polyhedra, starting with Archimedean polyhedral networks, consisting of sets of identical nodes, for which solutions have not previously been published.

We present a novel extension of the symmetry method to the solution of Catalan polyhedral networks, which have two or three different types of nodes, by superposition of partial solutions. We further show the usefulness of the symmetry method in simplifying problems by solving networks of four-dimensional polytopes, including the impressively complicated 120-cell. Finally, we derive a general solution for the n-dimensional hypercube using the symmetry method.

We confirmed our computed results with physical models of several of these networks, including the rhombic dodecahedron, the 4-simplex, and the 4-dimensional hypercube (tes-

saract). Using custom printed circuit boards and 0.1% precision resistors, we constructed equivalent networks and measured the resistance between nodes, obtaining values within 0.1% of our computed values.

1 Introduction

"Now we may ask an amusing question: What would happen if in the network of Figure [1.1] we kept on adding sections *forever*—as we indicate by the dashed lines in Figure [1.2]? Can we solve such an infinite network? Well, that's not so hard. First we notice that such an infinite network is unchanged if we add one more section to the 'front' end. Surely, if we add one more section to an infinite network it is still the same infinite network."

– R.P. Feynman [1]

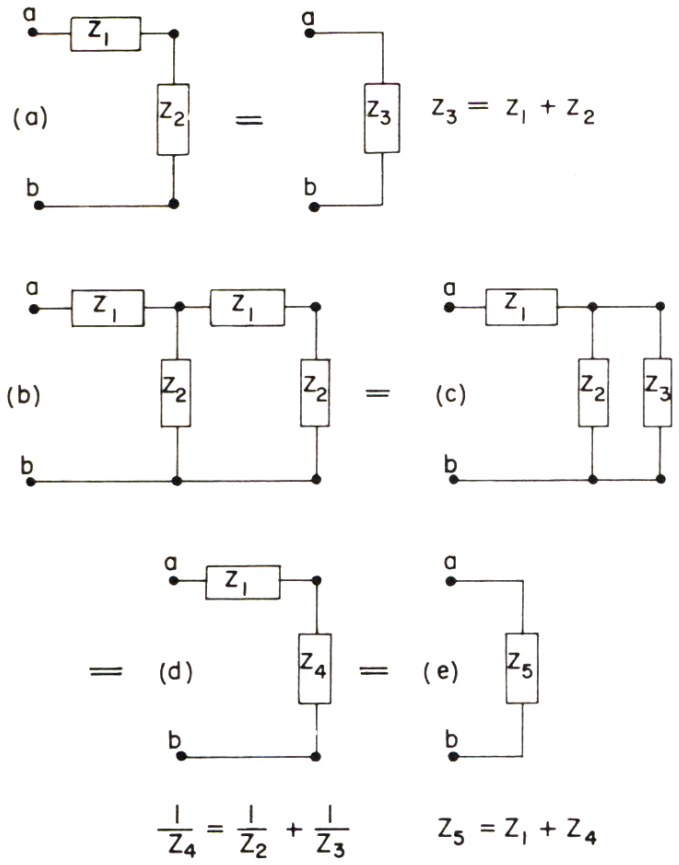

Figure 1.1: The effective impedance of a ladder, from [1]

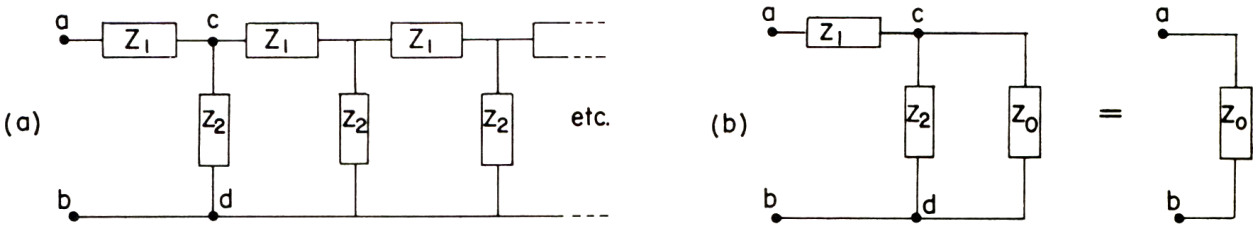

Figure 1.2: The effective impedance of an infinite ladder, from [1]

Richard Feynman was one of many physicists, mathematicians, and engineers who have studied networks of resistors. The infinite resistor ladder problem was already a well-known curiosity when he included it in his famed freshman physics lectures at Caltech in 1963. His solution:

$$z_0 = \frac{z_1}{2} + \sqrt{(\frac{{z_1}^2}{4}) + z_1 z_2} \tag{1.1}$$

hints at both unexpected mathematical beauty (if $z_1 = z_2 = 1$, then $z_0 = (1 + \sqrt{5})/2 = \phi$, the golden ratio) and a surprising practical application (if $2z_1 = z_2$, then $z_0 = z_2$, so such a ladder can be tapped at each "rung" to obtain a voltage divider that can be used as the basis for binary digital-to-analog or analog-to-digital conversion).

Feynman's method for solving the infinite resistor ladder is related to a method used in this project for solving more complex but finite resistor networks: both gain high efficiency by exploiting repetitive structures in networks to reduce the size of the problems to be solved.

In recent years, much of the research involving resistor networks has made use of Monte Carlo methods. If we consider the path taken by an electron through a network as a random walk in which the probability of exiting a node via a given edge is proportional to the potential difference between the edge's endpoints divided by the resistance along that edge (Ohm's law), it is apparent that any resistor network can be solved approximately by simulating the paths of many electrons and observing their flow. This approach has been applied to problems in materials science, traffic (and other network) congestion, percolation, and vascular flow, in which current flow through random resistor networks is a useful model.

Accounts of some of these developments can be found in [2] and [3].

This project, however, addresses exact methods for solving symmetric networks. The resistance between arbitrary nodes of a resistor network having the connectivity of one of the Platonic solids has been analyzed by van Steenwijk [4] by exploiting the symmetries of the structures. This project generalizes his approach and applies it to a much wider range of structures, including Archimedean and Catalan polyhedral and the higher dimensional regular polytopes. The great advantage of this approach is that it allows the problems to be solved without the enormous amount of computation that would be involved in a brute force approach. The simplification comes about because we exploit the symmetries of the structures to cut down considerably on the number of simultaneous equations that have to be solved. Most of the results we present here are new and have not previously been reported in the literature.

A problem related to the one investigated in this project is the resistance between arbitrary nodes of infinite lattices of various kinds. This problem has been investigated for a variety of three- and higher dimensional lattices in [5], [6], [7], [8], [9], [10].

Although the bulk of this project consists of theoretical work, we have actually constructed models of some of the networks and verified our predictions of the effective resistances in them.

2 Automatically Generating Kirchhoff Equations From an Arbitrary Network

A naïve approach to calculating the equivalent resistance in a network of resistors is to set up a system of linear equations based on Kirchhoff's laws for the network with an outside current passed between the two nodes of interest, and then to solve for the currents through each edge. Kirchhoff's current law states that the sum of currents flowing into a node is equal to the sum of currents exiting the node. Kirchhoff's voltage law states that sum of potential differences in any closed loop is zero. This project investigates networks with tens or hundreds of equations. For this reason, we developed methods for automatic generation of the Kirchhoff Equations. These equations can be easily solved with standard linear algebra packages, such as Maple. This allows us to quickly find the resistance between any two nodes of an arbitrary network.

There are two types of equations we need to generate for any given network. The first of these, the "node" equations, are defined by Kirchhoff's Current Law: for each node in the network, the sum of the currents entering the node is equal to the sum of the currents exiting the node. These equations are trivial to set up automatically, as the algorithm only needs to look at the immediate neighbors of each node. The second type, the "loop" equations, are defined by Kirchhoff's Voltage Law, which requires that the sum of the voltages around any closed loop be zero. These are not so simple to generate, as the algorithm must find closed loops within the network. Additionally, the algorithm should ideally generate only as many loop equations as are required to completely define our system and not waste time finding loops that are combinations of already included loops.

To solve this problem, we must generate a basis for the cycle space of our network. To do this, we first create a spanning tree using a breadth-first search of the network from the starting node (any spanning tree will work, but this is a convenient way to construct one). Each edge not in this spanning tree will be part of a distinct cycle in the network. To

generate the full loop from one of these edges, we start at the two endpoints (nodes) of that edge, and follow the edges going to parent nodes from our spanning tree until we reach a common ancestor node. Since we used a breadth first search to create the tree, we know the depth of all the nodes in our network (the minimum number of edges traversed to reach that node from the starting node), and we know that all the edges not in the spanning tree have endpoints with depths that differ by no more than one. This means that to find the nearest common ancestor in the tree, we only need to compare the pairs of equal depth ancestors of the two nodes until we find the same ancestor. This algorithm does not necessarily find the smallest loops, but it does guarantee that none of the loops found will be combinations of previously found loops, and that there will be no other loops in the network that are not combinations of found loops. To see this, note that any network with H nodes and E edges will have a cycle space of dimension $E - H + 1$. The spanning tree contains $H - 1$ edges, leaving exactly $E - H + 1$ edges not in the spanning tree. For each of these edges, we have created one and only one cycle containing that edge. We therefore have $E - H + 1$ independent cycles, and thus have a basis for the cycle space.

3 Symmetry Method

3.1 Introduction

The symmetry of the equal-resistor networks we investigated allows for a clever simplification of the problem of calculating the effective resistances. The idea, introduced by van Steenwijk [4] is to model the network when a current I is passed into one node and a current of $I/(H-1)$ is taken out through the remaining $H-1$ nodes. We then solve that system for the currents through all edges and superimpose it on the same network with all currents negated and rotated such that the current I now exits a node of interest. The superimposed system will be one in which current $HI/(H-1)$ enters one node and leaves another, and zero current enters or exits each other node.

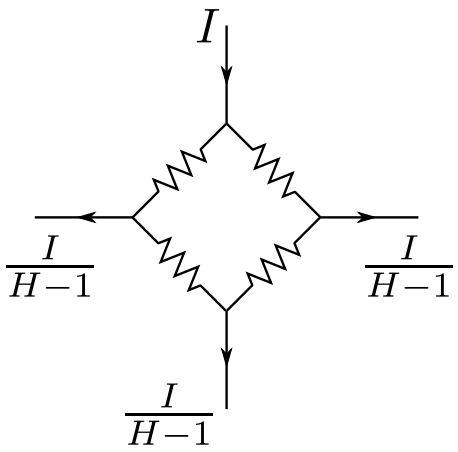

Figure 3.1: Example Network

This approach has two major benefits. First, since our superimposed system consists of two networks differing only by rotation, we need only solve the original system of equations once to obtain the resistance between any two nodes in the network. Second, we can look at the network as having layers of symmetrically equivalent nodes. Each of the symmetrically equivalent nodes will give rise to identical circuit equations, so it suffices to consider just a single node from a layer in setting up the equations. This leads to a significant reduction in the number of equations that have to be solved.

3.2 Notation

From a network we choose a starting node S, into which the external current I is passed. We form a layer by first picking a node W in the network not already assigned to a layer, and then finding all other nodes that, with respect to S, are symmetrically equivalent to W.

Formally, we define a 'layer' given any node W as the set of all nodes W_i, such that there exists an isomorphism f of the network with $f(S) = S$, and $f(W) = W_i$. To keep notation simple, we number these layers in order of their distance from S; however, any numbering would work. We then define a layer matrix L, for a network with l layers as an $l \times l$ matrix such that

$$L_{i,j} = \text{number of nodes in layer } j \text{ connected to any one node in layer } i.$$

We then define $I_{i,j}$ as the current passing from any node in layer i to a node in layer j. Again, since the layers are composed of nodes in the orbit of the isomorphism group that leaves S fixed, all edges connecting a node in layer i to a node in layer j will carry the same current. It is also important to note the direction of the current. While setting up our equations, we may not know which direction current will pass through an edge. To stay consistent, we set up equations such that current passes from a lower numbered layer to a higher numbered layer. If the current in fact flowed in the opposite direction for some edge, we would simply obtain a negative value for the current through that edge. After solving our system of equations for all $I_{i,j}$, we can obtain the potential difference from the starting node S to any desired node T as follows:

$$\left(I_{S,n_1} + I_{n_1,n_2} + \cdots + I_{n_d,T}\right) R \tag{3.1}$$

where R is the resistance of a single resistor, and $I_{S,n_1}, I_{n_1,n_2}, \cdots, I_{n_d,T}$ are the currents flowing through edges on any arbitrarily chosen path from node S to node T. In the rotated system, the potential difference between the same two nodes is:

$$-\left(I_{T,n_d} + I_{n_d,n_{d-1}} + \cdots + I_{n_1,S}\right) R \tag{3.2}$$

Since $I_{a,b} = -I_{b,a}\ \forall\ a, b$, the expressions (3.1) and (3.2) are equal.

The potential difference of the superposition of the original and rotated, current-reversed systems is the sum of (3.1) and (3.2), or

$$2\left(I_{S,n_1} + I_{n_1,n_2} + \cdots + I_{n_d,T}\right) R \tag{3.3}$$

Since the superposed system has a current of $HI/(H-1)$ entering node S and exiting node T, the equivalent resistance between S and T is

$$R_n = 2\left(I_{S,n_1} + I_{n_1,n_2} + \cdots + I_{n_d,n}\right) R\frac{(H-1)}{HI} \tag{3.4}$$

4 Semi-Regular Polyhedra

4.1 Introduction

In this section we investigate specifically the Archimedean and Catalan solids. The methods we present can be applied to any polyhedron exhibiting a high degree of symmetry, however. An Archimedean solid is a convex polyhedron consisting of two or more types of regular polygonal faces meeting at identical nodes. That is to say, the polyhedron appears the same when viewed from any node. The Catalan solids are the duals of the Archimedean Solids. (The dual of a polyhedron P is the polyhedron D with nodes corresponding to the faces of P, with pairs of nodes in D adjacent where the corresponding faces in P are adjacent. The dual of D is the original polyhedron P.) Among the Platonic solids, the cube and octahedron are duals, the dodecahedron and icosahedron are duals, and the tetrahedron is self-dual.

Since the nodes of each Archimedean solid are identical, we can apply the symmetry method with no modification. The Catalan solids, however, have two or three distinct types of nodes. Because of this, a solution to the symmetry method with a starting node of one type cannot be transformed into a solution with a starting node of another type. Instead, we create multiple systems; one for each type of starting node S. To find the equivalent resistance between two nodes of different types, we make a superposition of the two systems corresponding to those types, with one rotated appropriately. In practice, this can be treated as applying equation (3.4) to both systems, and averaging the results. It is important to note that the layers in the two sets of equations may not match up, so we must be careful to make sure we choose layers corresponding to the same pair of nodes in both solutions.

4.2 Rhombic Dodecahedron

The rhombic dodecahedron is one of the simplest Catalan solids. To find the equivalent resistance between nodes using the symmetry method, we must solve two separate sets of

equations. One set is obtained by passing a current of I in through a node of degree four, and taking a current of $I/13$ out each of the remaining 13 nodes. This divides the graph into five layers of equipotential nodes. For each layer, we choose a node and create a Kirchhoff current equation to describe all the currents entering and leaving that node.

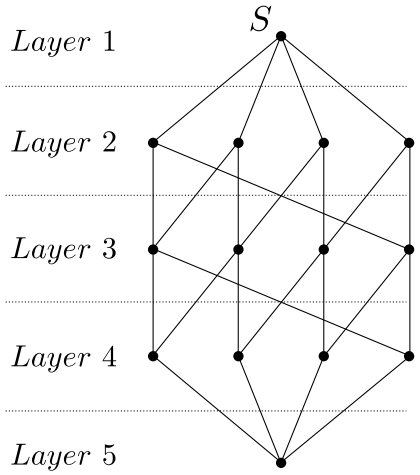

Figure 4.1: Layer diagram of the rhombic dodecahedron starting from a node of degree four

The first step is to set up the layer matrix. The layer diagram shows the starting node (layer 1) connected to four nodes in layer 2 and to no nodes in any other layer, hence the first row of the matrix is:

$$\begin{bmatrix} 0 & 4 & 0 & 0 & 0 \end{bmatrix}$$

We then choose any one node in layer 2, and see that it is connected to one node in layer 1 and two nodes in layer 3, making the second row:

$$\begin{bmatrix} 1 & 0 & 2 & 0 & 0 \end{bmatrix}$$

The remaining rows follow similarly, making the full layer matrix:

$$L_4 = \begin{bmatrix} 0 & 4 & 0 & 0 & 0 \\ 1 & 0 & 2 & 0 & 0 \\ 0 & 2 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 1 \\ 0 & 0 & 0 & 4 & 0 \end{bmatrix} \tag{4.1}$$

Next, we use the layer matrix to set up the current equations:

$$\begin{aligned} I - 4I_{1,2} &= 0 \\ I_{1,2} - 2I_{2,3} - \frac{I}{13} &= 0 \\ 2I_{2,3} - 2I_{3,4} - \frac{I}{13} &= 0 \\ 2I_{3,4} - I_{4,5} - \frac{I}{13} &= 0 \\ 4I_{4,5} - \frac{I}{13} &= 0 \end{aligned} \tag{4.2}$$

We then repeat the process with the current I now entering a node of degree three:

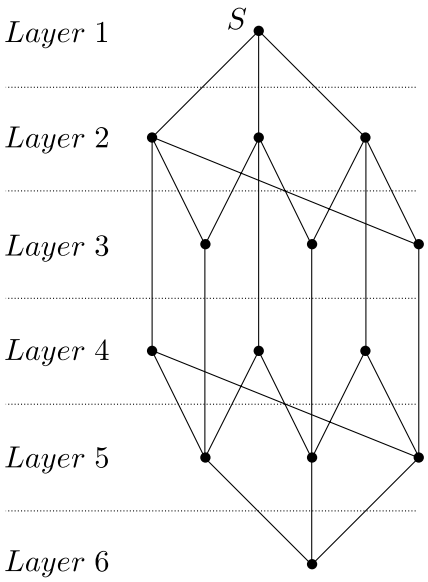

Figure 4.2: Layer diagram of the rhombic dodecahedron starting from a node of degree three

The layer matrix and node equations for the currents (now denoted J) take the form:

$$L_3 = \begin{bmatrix} 0 & 3 & 0 & 0 & 0 & 0 \\ 1 & 0 & 2 & 1 & 0 & 0 \\ 0 & 2 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 2 & 0 \\ 0 & 0 & 1 & 2 & 0 & 1 \\ 0 & 0 & 0 & 0 & 3 & 0 \end{bmatrix} \tag{4.3}$$

$$\begin{aligned} I - 3J_{1,2} &= 0 \\ J_{1,2} - 2J_{2,3} - J_{2,4} - \frac{I}{13} &= 0 \\ 2J_{2,3} - J_{3,5} - \frac{I}{13} &= 0 \\ J_{2,4} - 2J_{4,5} - \frac{I}{13} &= 0 \\ J_{3,5} + 2J_{4,5} - J_{5,6} - \frac{I}{13} &= 0 \\ 3J_{5,6} - \frac{I}{13} &= 0 \end{aligned} \tag{4.4}$$

Here, there are six equations and six unknown variables, however, one of the equations is redundant, so we do not have enough to solve the system. We fix that by writing a Kirchhoff voltage equation for the two distinct paths from layer 2 to layer 5 in figure 4.2: $2 \to 3 \to 5$ and $2 \to 4 \to 5$. This gives the equation:

$$J_{2,3} + J_{3,5} - J_{2,4} - J_{4,5} = 0 \tag{4.5}$$

We now have enough equations to solve both systems to obtain the following values:

$$I_{1,2} = \frac{1}{4}I \qquad I_{2,3} = \frac{9}{104}I \qquad I_{3,4} = \frac{5}{104}I \qquad I_{2,3} = \frac{1}{52}I \tag{4.6}$$

$$J_{1,2} = \frac{1}{3}I \qquad J_{2,3} = \frac{11}{156}I \qquad J_{2,4} = \frac{3}{26}I \qquad J_{3,5} = \frac{5}{78}I \qquad J_{4,5} = \frac{1}{52}I \qquad J_{5,6} = \frac{1}{39}I \tag{4.7}$$

To obtain the resistance between two nodes of degree four, we use equation (3.4) with the values obtained in (4.6) :

$$\begin{aligned} R_{4,3} &= (I_{1,2} + I_{2,3})\,\frac{26}{14I} = \frac{5}{8} \\ R_{4,5} &= (I_{1,2} + I_{2,3} + I_{3,4} + I_{4,5})\,\frac{26}{14I} = \frac{3}{4} \end{aligned} \tag{4.8}$$

where $R_{a,b}$ represents the resistance from a starting node of degree a to a node in layer b of the corresponding layer diagram.

We obtain the resistance between two nodes of degree three similarly, using the values obtained in (4.7):

$$\begin{aligned} R_{3,3} &= (J_{1,2} + J_{2,3})\,\frac{26}{14I} = \frac{3}{4} \\ R_{3,4} &= (J_{1,2} + J_{2,4})\,\frac{26}{14I} = \frac{5}{6} \\ R_{3,6} &= (J_{1,2} + J_{2,3} + J_{3,5} + J_{5,6})\,\frac{26}{14I} = \frac{11}{12} \end{aligned} \tag{4.9}$$

To obtain the resistance between a node of degree three and a node of degree four, we average the values obtained by equation (3.4) with (4.6), and (4.7):

$$\begin{aligned} R_{4,2} = R_{3,2} &= (I_{1,2} + J_{1,2})\,\frac{13}{14I} = \frac{13}{24} \\ R_{4,4} = R_{3,5} &= (I_{1,2} + I_{2,3} + I_{3,4} + J_{1,2} + J_{2,3} + J_{3,5})\,\frac{13}{14I} = \frac{19}{24} \end{aligned} \tag{4.10}$$

5 The 120 Cell

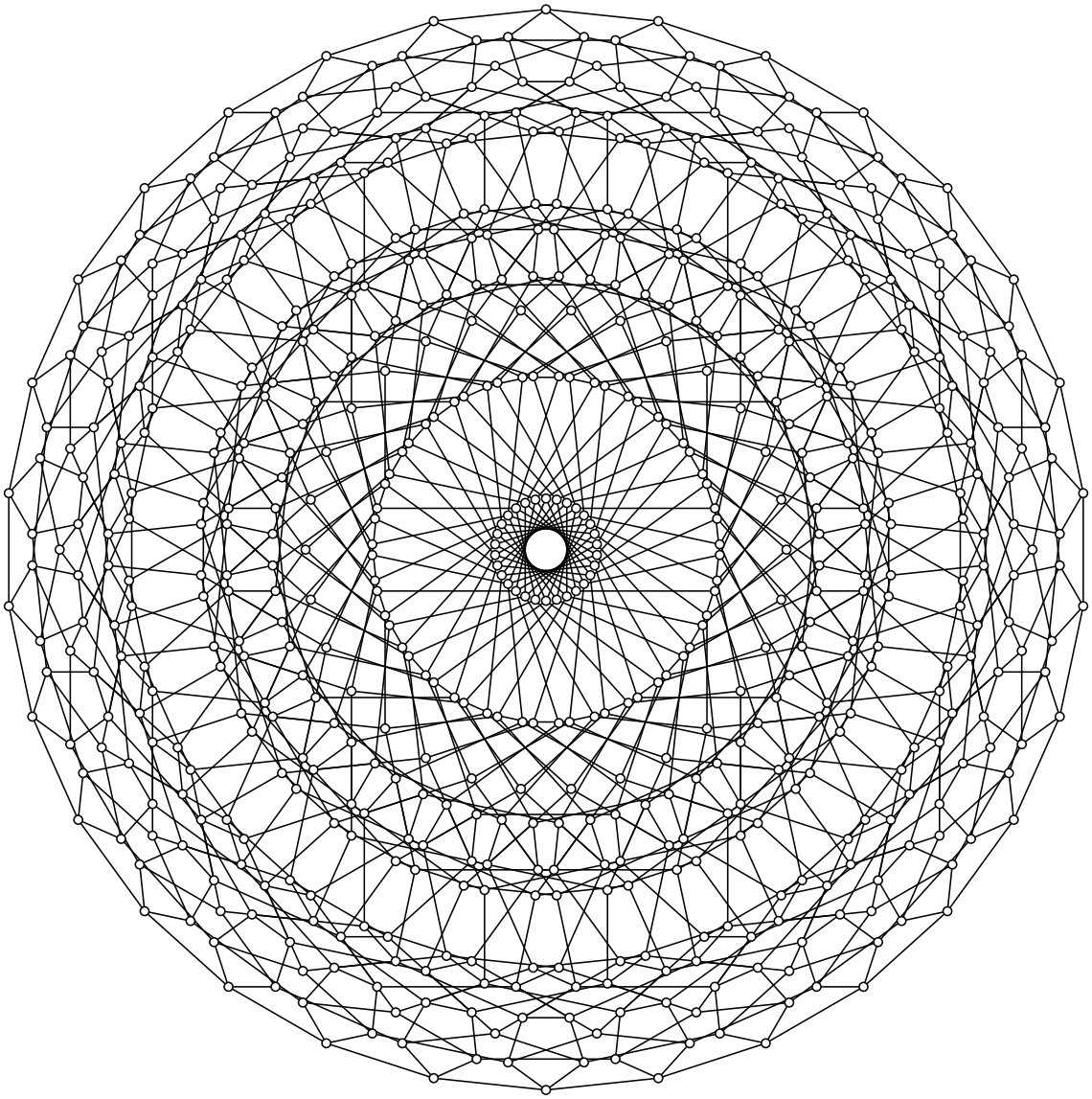

Figure 5.1: 120-cell

With 1200 edges and 45 node layers, the 120-cell is the most complex of the regular polychora, and is an excellent example for showing the power of the symmetry method. The naïve approach described in Section 2 would require solving a system of 1200 variables for each pair of nodes to be examined. The symmetry method, however, only requires solving a

system of 61 variables, and this system does not need to be modified for each choice of current-entering and exiting nodes. The regular 120-cell also has the unique property among four and lower dimensional regular convex polytopes of having nodes from different layers at the same Euclidean distance from the starting node. This first occurs for layers 8 and 9: for the regular 120-cell of radius 1, each of these layers is composed of 12 nodes at a distance of $\sqrt{11/8}$ from the starting node [11]. Remarkably, despite not having the same number of connections to other layers, the equivalent resistance from the starting node to any node in either of these layers is the same. Layers 10, 11, and 12 are also equidistant from the starting node, but layers 10 and 11 consist of four nodes each, while layer 12 consists of 24 nodes. We again find the equivalent resistance to layer 10 and 11 to be the same, while the resistance to layer 12 differs. In fact, every pair of layers at the same Euclidean distance to the starting node and with the same number of nodes within the layer have the same resistance. The layers along with resistance values are summarized in table 5.1.

Layer	Coxeter Section*	Number of nodes	Connections to Other Layers**	Resistance From Starting Node	Resistance (Decimal)
1	0	1	2, 2, 2, 2	0	0
2	1	4	1, 3, 3, 3	599/1200	0.499167
3	2	12	2, 3, 4, 4	299/450	0.664444
4	3	24	3, 4, 5, 7	1789/2400	0.745417
5	4	12	4, 4, 6, 8	429132199/554677200	0.773661
6	5	4	5, 5, 5, 11	48490591/61630800	0.786792
7	6	24	4, 7, 9, 10	881727163/1109354400	0.794811
8	7	12	5, 10,10, 13	225666647/277338600	0.813686
9	7	12	7, 7, 12, 15	225666647/277338600	0.813686
10	8	24	7, 8, 14, 16	455942897/554677200	0.821997

11	8	4	6, 13, 13, 13	30427717/36978480	0.822849
12	8	4	9, 9, 9, 18	30427717/36978480	0.822849
13	9	12	8, 11, 17, 17	462466259/554677200	0.833757
14	9	12	10, 10, 14, 20	462466259/554677200	0.833757
15	10	12	9, 16, 16, 19	77556889/92446200	0.838941
16	11	24	10, 15, 17, 21	34611817/41087200	0.842399
17	12	24	13, 16, 17, 24	940267063/1109354400	0.847580
18	12	4	12, 19, 19, 19	78302369/92446200	0.847005
19	13	12	15, 18, 22, 22	43060369/50425200	0.853945
20	13	12	14, 21, 21, 23	43060369/50425200	0.853945
21	14	24	16, 20, 22, 25	951542863/1109354400	0.857745
22	15	24	19, 21, 22, 29	239412941/277338600	0.863251
23	15	6	20, 20, 26, 26	47879819/55467720	0.863201
24	15	24	17, 24, 25, 27	239412941/277338600	0.863251
25	16	24	21, 24, 26, 30	137567809/158479200	0.868050
26	17	12	23, 25, 25, 32	483007859/554677200	0.870791
27	17	12	24, 24, 28, 31	483007859/554677200	0.870791
28	18	4	27, 27, 27, 34	80877569/92446200	0.874861
29	18	24	22, 29, 30, 33	970386463/1109354400	0.874731
30	19	24	25, 29, 31, 36	671491/765600	0.877078
31	20	12	27, 30, 30, 37	81210889/92446200	0.878466
32	21	12	26, 32, 36, 36	488409659/554677200	0.880530
33	21	12	29, 29, 35, 38	488409659/554677200	0.880530
34	22	4	28, 37, 37, 37	32679277/36978480	0.883738
35	22	4	33, 33, 33, 40	32679277/36978480	0.883738
36	22	24	30, 32, 38, 39	21312439/24116400	0.883732

37	23	12	31, 34, 39, 39	35086721/39619800	0.885586
38	23	12	33, 36, 36, 41	35086721/39619800	0.885586
39	24	24	36, 37, 39, 42	985552963/1109354400	0.888402
40	25	4	35, 41, 41, 41	18284397/20543600	0.890029
41	26	12	38, 40, 42, 42	70603657/79239600	0.891015
42	27	24	39, 41, 42, 43	1264573/1416800	0.892556
43	28	12	42, 42, 43, 44	356602/398475	0.894917
44	29	4	43, 43, 43, 45	634943/708400	0.896306
45	30	1	44, 44, 44, 44	9533/10626	0.897139

Table 5.1: Resistance of the 120-Cell

*H.S.M. Coxeter summarizes the nodes of the regular 120-cell in 31 sections, based on their Euclidean distance from a starting node. Since some sections contain multiple layers, the section number is not enough to define all layers. The section numbers, however, are included as a convenience to understanding the geometry of the 120-cell.

**Rather than include the entire (45×45) layer matrix, the connectivity information for the 120-cell is summarized by lists of connected layers. Each node in the 120-cell is adjacent to four other nodes. We can therefore use the simple notation of listing the layer numbers of the four nodes connected to any one node in a given layer. For instance, layer 3 is listed as having connections to 2, 3, 4, 4. This means that each of the 12 nodes in layer 3 is connected to one node in layer 2, one node in layer 3, and two nodes in layer 4. The third row of the layer matrix is therefore:

$$\begin{bmatrix} 0 & 1 & 1 & 2 & 0 & 0 & \cdots \end{bmatrix}$$

6 N-Dimensional Hypercube

The symmetry method introduced in chapter 3 can also be used to determine the resistance between any two nodes of an N-dimensional hypercube. Each node in layer k is adjacent to $k-1$ nodes in layer $k-1$ and $N-k+1$ nodes in layer $k+1$. Since there are no connections between nodes more than one layer apart, the currents between layers can be found by the simple recursion relation:

$$\begin{aligned} I_{N,1} &= \frac{1}{N} \\ I_{N,k} &= \frac{1}{N-k+1}\left[I_{N,k-1}(k-1) - \frac{1}{2^N-1}\right] \end{aligned} \tag{6.1}$$

Using these currents with equation (3.4), we can find $R_{N,m}$, the resistance between two nodes on an N-dimensional hypercube m edges apart

$$R_{N,m} = \frac{2^{N+1}-2}{2^N}\sum_{k=1}^{m} I_{N,k} \tag{6.2}$$

The first several values of $R_{N,m}$ are given in the table below:

Distance	1D	2D	3D	4D	5D	6D	7D	8D	9D
1	1	$\frac{3}{4}$	$\frac{7}{12}$	$\frac{15}{32}$	$\frac{31}{80}$	$\frac{21}{64}$	$\frac{127}{448}$	$\frac{255}{1024}$	$\frac{511}{2304}$
2		1	$\frac{3}{4}$	$\frac{7}{12}$	$\frac{15}{32}$	$\frac{31}{80}$	$\frac{21}{64}$	$\frac{127}{448}$	$\frac{255}{1024}$
3			$\frac{5}{6}$	$\frac{61}{96}$	$\frac{241}{480}$	$\frac{131}{320}$	$\frac{12}{35}$	$\frac{2105}{7168}$	$\frac{16531}{64512}$
4				$\frac{2}{3}$	$\frac{25}{48}$	$\frac{101}{240}$	$\frac{7}{20}$	$\frac{167}{560}$	$\frac{929}{3584}$
5					$\frac{8}{15}$	$\frac{137}{320}$	$\frac{2381}{6720}$	$\frac{10781}{35840}$	$\frac{42061}{161280}$
6						$\frac{13}{30}$	$\frac{343}{960}$	$\frac{2033}{6720}$	$\frac{9383}{35840}$
7							$\frac{151}{420}$	$\frac{32663}{107520}$	$\frac{84677}{322560}$
8								$\frac{32}{105}$	$\frac{2357}{8960}$
9									$\frac{83}{315}$

Table 6.1: Resistance of the N-dimensional hypercube

The calculated resistances on the principal diagonal of table 6.1 are in agreement with the previously published results of Huang and Tretiak [12].

7 Physical Models

For the last part of our project, we wanted to make physical models of some of the networks to compare the actual measured resistance to that of our theoretical models. We wanted high precision measurements, so we used 1kΩ ±0.1% tolerance resistors soldered to printed circuit boards. Figure 7.1 below shows the PCB we used. The contacts are connected in rows by copper traces. Since the traces have significantly lower resistance than the resistors, we can treat each row as a node in the network. Resistors are attached to contacts between two rows of the PCB whenever there is an edge between the corrisponding nodes of the network

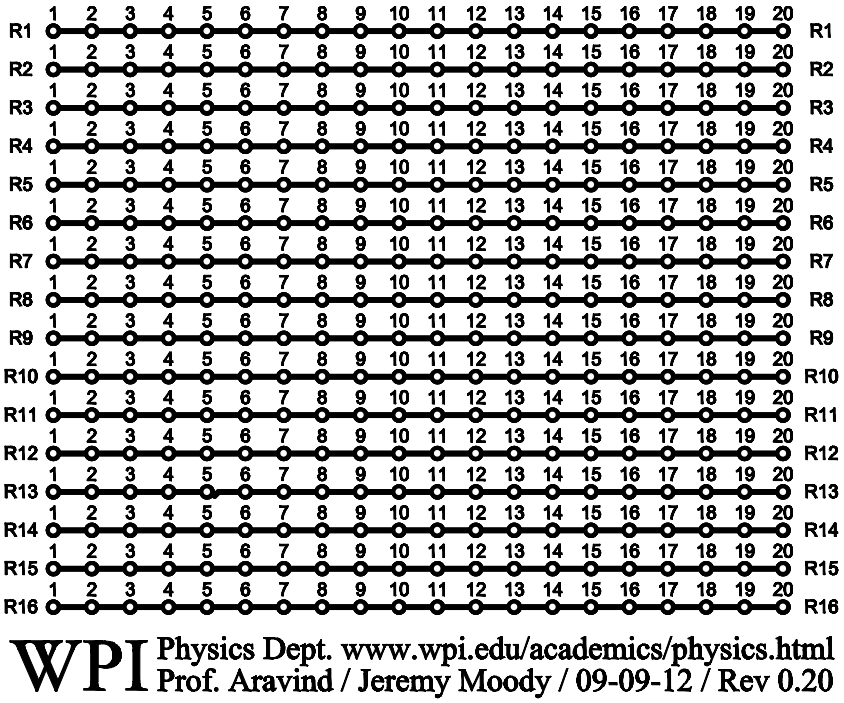

Figure 7.1: Printed circuit board diagram

We used a high precision ohmmeter to measure the resistance between two rows. The measured and calculated values, expressed as a fraction of the edge resistance, are summarized in tables 7.1 to 7.4.

Distance	Measured	Calculated
1	.40001	.40000

Table 7.1: Resistance of the 4-simplex

Distance	Measured	Calculated
1	.29162	.29167
2	.33330	.33333

Table 7.2: Resistance of the 16-cell

Distance	Measured	Calculated
1	.46878	.46875
2	.58335	.58333
3	.63542	.63542
4	.66666	.66667

Table 7.3: Resistance of the 4-cube

Distance	Measured	Calculated
Between nodes of degree 3		
2	.74989	.75000
4	.83317	.83333
6	.91644	.91666
Between nodes of degree 4		
2	.62495	.62500
4	.74992	.75000
Between nodes of degree 3 and 4		
1	.54163	.54167
3	.79149	.79167

Table 7.4: Resistance of the rhombic dodecahedron

Some of our measured values match the calculated values to the precision of the ohmmeter we used. The worst measured value differed from the theoretical value by less than 0.03%.

Since this is better than the manufactured tolerance of the resistors we used, we can assert that our physical and theoretical models are accurate.

8 Conclusions and Further Study

In this project, we developed novel, efficient, and exact methods for determining the effective resistance between arbitrary nodes in a variety of symmetric networks of equal resistors, including several types of networks for which no solutions have previously been published.

An intriguing question that arose from this project is: Are the equivalent resistances between a node S and T equal, for all T belonging to layers of equal multiplicity and Euclidean distance from S, if S and T are nodes of a regular polytopal equal-resistor network?

We have only discussed a few cases in te text, but the resistances in many of the other structures we studied can be found in Appendix C.1.

We observed this to be true in the 120-cell, the only regular convex polytope of dimension 4 or fewer that has such layers. We conjecture that this property might be shared by higher-dimensional regular polytopes.

While the circuits based on the 600-cell and 120-cell might not be easy to construct, it might be a useful exercise to calculate the resistances in these structures using the Monte Carlo method and to corroborate the results obtained here.

Although we investigated a limited number of networks in detail, the modifications we made to the symmetry method allow it to be used to simplify calculations of resistance on any network exhibiting symmetry. We constructed physical models of several of these networks and validated our methods by comparing our calculations with direct measurements of the physical models.

References

[1] R. P. Feynman, R. B. Leighton, and M. Sands. *The Feynman Lectures on Physics*, volume II. Addison-Wesley, 1964.

[2] P.J. Nahin. *Mrs. Perkins's Electric Quilt: And Other Intriguing Stories of Mathematical Physics*. Princeton University Press, 2009.

[3] P.G. Doyle and J.L. Snell. *Random walks and electric networks*. Carus mathematical monographs. Mathematical Association of America, 1984.

[4] F. J. van Steenwijk. Equivalent resistors of polyhedral resistive structures. *American Journal of Physics*, 66(1):90–91, January 1998.

[5] József Cserti. Application of the lattice Green's function for calculating the resistance of an infinite network of resistors. *American Journal of Physics*, 68(10):896, October 2000.

[6] D. Atkinson and F. J. van Steenwijk. Infinite resistive lattices. *American Journal of Physics*, 67(6):486, June 1999.

[7] Monwhea Jeng. Random walks and effective resistances on toroidal and cylindrical grids. *American Journal of Physics*, 68(1):37, January 2000.

[8] Raymond A. Sorensen. The random walk method for DC circuit analysis. *American Journal of Physics*, 58(11):1056, November 1990.

[9] Leo Lavatelli. The resistive net and finite-difference equations. *American Journal of Physics*, 40(9):1246, September 1972.

[10] Giulio Venezian. On the resistance between two points on a grid. *American Journal of Physics*, 62(11):1000, September 1994.

[11] H.S.M. Coxeter. *Regular Polytopes*. Dover books on advanced mathematics. Dover Pub., 1973.

[12] T. S. Huang and O. J. Tretiak. Resistance of an n-dimensional cube. *Proceedings of the IEEE*, 53:1271–1272, September 1965.

[13] Antoni Amengual. The intriguing properties of the equivalent resistances of n equal resistors combined in series and in parallel. *American Journal of Physics*, 68(2):175, February 2000.

Appendices

A Code For Generating Kirchhoff Equations

```
import java.util.Scanner;
import java.util.Queue;
import java.util.LinkedList;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.Reader;
import java.util.StringTokenizer;
public class Maplegenerator {

    public static void main(String[] args){  //args[0] is the path to the (.csv) file containing the network
        int[] nodedepth,lowerN;
        int[][] network;
        Queue<Integer> q = new LinkedList();
        Scanner scan = new Scanner(System.in);
        int n, i, j, k, l, m, p, numeq, currentnode, maxdepth =0;
        boolean first = true, ndone;
        network = getArray(args[0]);
        n = network.length;
        lowerN = new int[n];

        System.out.println("Network Matrix:");
        for(i=0;i<n;i++){
            for(j=0;j<n;j++){
                if(network[i][j] == -1) System.out.print("-1 ");
                else  System.out.print(" " + network[i][j] + " ");
            }
            System.out.println();
        }

/** Begin Maple code generation **/
        System.out.println("\n\n\nrestart;");

/** Node Equations **/
        for(i=0;i<n;i++){
            System.out.print("e"+i+" :=");
            for(j=0;j<n;j++){
                if(network[i][j] == 1)
                    System.out.print(" + i" + i + "t" + j);
                else if(network[i][j] == -1)
                    System.out.print(" - i" + j + "t" + i);
```

```
            }
            if(i==0)
                System.out.println(" = 1:");
            else if(i==1)
                System.out.println(" = -1:");
            else
                System.out.println(" = 0:");
        }

        System.out.println();

/** Loop Equations **/
        numeq = n-1;

/** First number the nodes of the matrix by distance from node zero using a breadth first search **/
        nodedepth = new int[n];
        for(i=1;i<n;i++){
            nodedepth[i]=-1;
        }
        nodedepth[0] = 0;
        q.add(0);
        while(!q.isEmpty()){
            currentnode=q.remove();
            for(i=1;i<n;i++){
                if(network[i][currentnode] != 0 && nodedepth[i] == -1){
                    nodedepth[i] = nodedepth[currentnode] + 1;
                    lowerN[i] = currentnode;                                    //Remember the parent of each node for easy loop construction.
                    if(nodedepth[i]>maxdepth){
                        maxdepth = nodedepth[i];
                    }
                    q.add(i);
                }
            }
        }

        for(j=1;j<n;j++){
            i = nodedepth[j];
            for(k=0;k<n;k++){
                if(network[j][k] != 0 && nodedepth[k]==i-1){     //find all nodes connected to node "j" at level i-1.
                    for(l=k+1;l<n;l++){                  //Find all nodes greater than "k" connected to node "j" at level i-1.
                        if(network[j][l] != 0 && nodedepth[l]==i-1){
                            numeq++;
                            System.out.print("e" + numeq + " := ");
                            if(j>k) System.out.print("-i" + k + "t" + j);
                            else System.out.print("i" + j + "t" + k);
                            if(j>l) System.out.print(" + i" + l + "t" + j);
                            else System.out.print(" - i" + j + "t" + l);
                            ndone = true;
                            int k1 = k,l1 = l ;
```

```
                        while(ndone){
                            if(k>lowerN[k1]) System.out.print(" - i" + ↩
                               lowerN[k1] + "t" + k1);
                            else System.out.print( " + i" + (k1+1) + "t"↩
                                + lowerN[k1]);
                            if(l1>lowerN[k1]) System.out.print(" + i" + ↩
                               lowerN[l1] + "t" + l1);
                            else System.out.print(" - i" + (l1+1) + "t" ↩
                               + lowerN[l1]);
                            if(lowerN[k1] == lowerN[l1])
                                ndone = false;
                            else{
                                k1=lowerN[k1];
                                l1=lowerN[l1];
                            }
                        }
                        System.out.println(" = 0:");
                    }
                }
                break;
            }
        }

        for(k=j+1;k<n;k++){
            if(network[j][k] != 0 && nodedepth[k]==i){  //find all nodes↩
                greater than j connected to "j" at level i.
                numeq++;
                System.out.print("e" + numeq + " := i" + j + "t" + k);
                ndone = true;
                int j1 = j,k1 = k ;
                while(ndone){
                    if(k1>lowerN[k1]) System.out.print(" - i" + lowerN[↩
                       k1] + "t" + k1);
                    else System.out.print( " + i" + k1 + "t" + lowerN[k1↩
                       ]);
                    if(j1>lowerN[j1]) System.out.print(" + i" + lowerN[↩
                       j1] + "t" + j1);
                    else System.out.print(" - i" + j1 + "t" + lowerN[j1↩
                       ]);
                    if(lowerN[k1] == lowerN[j1])
                        ndone = false;
                    else{
                        k1=lowerN[k1];
                        j1=lowerN[j1];
                    }
                }
                System.out.println(" = 0:");
            }
        }
    }

/** Solve, Assign **/
    System.out.print("s := solve({");
    for(i=0;i<numeq;i++){
```

```
            System.out.print("e" + i + ", ");
        }
        System.out.print("e" + numeq + "},{");
        for(i=0;i<n;i++){
            for(j=i+1;j<n;j++){
                if(network[i][j]==1)
                    if(first){
                        System.out.print("i" + i + "t" + j);
                        first=false;

                    }
                    else
                        System.out.print(", i" + i + "t" + j);
            }
        }
        System.out.print("});\nassign(s):");
    }

    public static int[][] getArray (String file){
        int[][] network = null;
        int size = 0;
        try{
            BufferedReader br = new BufferedReader(new FileReader(file));
            String line = br.readLine();
            int row = 0;
            int col = 0;
            StringTokenizer st = new StringTokenizer(line,",");
            while (st.hasMoreTokens())
            {
                st.nextToken();
                size++;
            }
            network = new int[size][size];

            while(line != null)
            {
                StringTokenizer st2 = new StringTokenizer(line,",");
                while (st2.hasMoreTokens())
                {
                    network[row][col] = Integer.parseInt(st2.nextToken());
                    col++;
                }
                col =0;
                row++;
                line = br.readLine();
            }
            br.close();
        }
        catch(Exception e){
        }
        return network;
    }
}
```

B Sample Input and Output

The following input for the code in Appendix A represents a Rhombic Dodecahedron.

```
RhombicD.csv
0,1,1,1,0,0,0,0,0,0,0,0,0,0
-1,0,0,0,1,1,0,0,0,0,0,0,1,0
-1,0,0,0,1,0,1,0,1,0,0,0,0,0
-1,0,0,0,0,1,1,0,0,0,1,0,0,0
0,-1,-1,0,0,0,0,1,0,0,0,0,0,0
0,-1,0,-1,0,0,0,0,0,0,0,1,0,0
0,0,-1,-1,0,0,0,0,0,1,0,0,0,0
0,0,0,0,-1,0,0,0,1,0,0,0,1,1
0,0,-1,0,0,0,0,-1,0,1,0,0,0,0
0,0,0,0,0,0,-1,0,-1,0,1,0,0,1
0,0,0,-1,0,0,0,0,0,-1,0,1,0,0
0,0,0,0,0,-1,0,0,0,0,-1,0,1,1
0,-1,0,0,0,0,0,-1,0,0,0,-1,0,0
0,0,0,0,0,0,0,-1,0,-1,0,-1,0,0
```

This results in the following output from the program:

```
Network Matrix:
 0  1  1  1  0  0  0  0  0  0  0  0  0  0
-1  0  0  0  1  1  0  0  0  0  0  0  1  0
-1  0  0  0  1  0  1  0  1  0  0  0  0  0
-1  0  0  0  0  1  1  0  0  0  1  0  0  0
 0 -1 -1  0  0  0  0  1  0  0  0  0  0  0
 0 -1  0 -1  0  0  0  0  0  0  0  1  0  0
 0  0 -1 -1  0  0  0  0  0  1  0  0  0  0
 0  0  0  0 -1  0  0  0  1  0  0  0  1  1
 0  0 -1  0  0  0  0 -1  0  1  0  0  0  0
 0  0  0  0  0  0 -1  0 -1  0  1  0  0  1
 0  0  0 -1  0  0  0  0  0 -1  0  1  0  0
 0  0  0  0  0 -1  0  0  0  0 -1  0  1  1
```

```
 0 −1  0  0  0  0  0 −1  0  0  0 −1  0  0
 0  0  0  0  0  0  0 −1  0 −1  0 −1  0  0
```

```
restart;
e0 := + i0t1 + i0t2 + i0t3 = 1:
e1 := − i0t1 + i1t4 + i1t5 + i1t12 = −1:
e2 := − i0t2 + i2t4 + i2t6 + i2t8 = 0:
e3 := − i0t3 + i3t5 + i3t6 + i3t10 = 0:
e4 := − i1t4 − i2t4 + i4t7 = 0:
e5 := − i1t5 − i3t5 + i5t11 = 0:
e6 := − i2t6 − i3t6 + i6t9 = 0:
e7 := − i4t7 + i7t8 + i7t12 + i7t13 = 0:
e8 := − i2t8 − i7t8 + i8t9 = 0:
e9 := − i6t9 − i8t9 + i9t10 + i9t13 = 0:
e10 := − i3t10 − i9t10 + i10t11 = 0:
e11 := − i5t11 − i10t11 + i11t12 + i11t13 = 0:
e12 := − i1t12 − i7t12 − i11t12 = 0:
e13 := − i7t13 − i9t13 − i11t13 = 0:

e14 := −i1t4 + i2t4 − i0t1 + i0t2 = 0:
e15 := −i1t5 + i3t5 − i0t1 + i0t3 = 0:
e16 := −i2t6 + i3t6 − i0t2 + i0t3 = 0:
e17 := −i4t7 − i7t8 − i1t4 + i2t8 − i0t1 + i0t2 = 0:
e18 := −i4t7 − i7t12 − i1t4 + i1t12 = 0:
e19 := −i6t9 + i8t9 − i2t6 + i2t8 = 0:
e20 := −i6t9 − i9t10 − i2t6 + i3t10 − i0t2 + i0t3 = 0:
e21 := −i5t11 + i10t11 − i1t5 + i3t10 − i0t1 + i0t3 = 0:
e22 := −i5t11 − i11t12 − i1t5 + i1t12 = 0:
e23 := −i7t13 + i9t13 − i4t7 + i6t9 − i1t4 + i2t6 − i0t1 + i0t2 = 0:
e24 := −i7t13 + i11t13 − i4t7 + i5t11 − i1t4 + i1t5 = 0:
```

```
s := solve({e0, e1, e2, e3, e4, e5, e6, e7, e8, e9, e10, e11, e12, e13, e14,↩
    e15, e16, e17, e18, e19, e20, e21, e22, e23, e24},{i0t1, i0t2, i0t3, ↩
   i1t4, i1t5, i1t12, i2t4, i2t6, i2t8, i3t5, i3t6, i3t10, i4t7, i5t11, i6t9↩
   , i7t8, i7t12, i7t13, i8t9, i9t10, i9t13, i10t11, i11t12, i11t13});
assign(s):
```

The output (starting at the word "restart;") is a Maple program, which solves for all currents in the network when a current of 1 enters node 0, and exits node 1, as seen in the first two node equations (e0 and e1). The currents entering or exiting nodes can be changed by modifying the right-hand-side of the node equations, and running the Maple program again. The potential difference between the current-entering and current-exiting nodes is the sum of the currents along any path between the two nodes, multiplied by the resistance of a single resistor (1 Ω). The resistance between the nodes is this potential divided by the entering and exiting current (1 A).

Given the code above, Maple outputs the following:

```
{i0t1 = 13/24, i0t2 = 11/48, i0t3 = 11/48, i10t11 = 1/16, i11t12 = 1/16, ↩
   i11t13 = −1/48, i1t12 = −1/8, i1t4 = −1/6, i1t5 = −1/6, i2t4 = 7/48, i2t6↩
    = 1/48, i2t8 = 1/16, i3t10 = 1/16, i3t5 = 7/48, i3t6 = 1/48, i4t7 = ↩
   −1/48, i5t11 = −1/48, i6t9 = 1/24, i7t12 = 1/16, i7t13 = −1/48, i7t8 = ↩
   −1/16, i8t9 = 0, i9t10 = 0, i9t13 = 1/24}
```

This shows the resistance between nodes 0 and 1 is 13/24 Ω.

C Resistance of Selected Networks

C.1 Regular Convex Polychora

Distance	Resistance (Exact)	Resistance (Decimal)
1	2/5	0.40000

Table C.1: 5-cell (simplex)

Distance	Resistance (Exact)	Resistance (Decimal)
1	15/32	0.46875
2	7/12	0.58333
3	61/96	0.63542
4	2/3	0.66667

Table C.2: 8-cell (4-cube)

Distance	Resistance (Exact)	Resistance (Decimal)
1	7/24	0.29167
2	1/3	0.33333

Table C.3: 16-cell

Distance	Resistance (Exact)	Resistance (Decimal)
1	23/96	0.23958
2	11/40	0.27500
3	139/480	0.28958
4	3/10	0.30000

Table C.4: 24-cell

Distance	Resistance (Exact)	Resistance (Decimal)
1	119/720	0.16528
2	14293/75600	0.18906
3	737/3780	0.19497
4	1903/9450	0.20138
5	37/180	0.20556
6	5231/25200	0.20758
7	3179/15120	0.21025
8	40/189	0.21164

Table C.5: 600-cell

C.2 Archimedean Solids

Distance	Resistance (Exact)	Resistance (Decimal)
1	11/24	0.45833
2	7/12	0.58333
3	5/8	0.62500
4	2/3	0.66667

Table C.6: Cuboctahedron

Distance	Resistance (Exact)	Resistance (Decimal)
1	29/60	0.48333
2	61/90	0.67778
2	127/180	0.70556
3	7/9	0.77778
3	49/60	0.81667
4	38/45	0.84444
4	157/180	0.87222
5	8/9	0.88889

Table C.7: Icosidodecahedron

C.3 Catalan Solids

Distance	Resistance (Exact)	Resistance (Decimal)
Between nodes of degree 3		
2	3/4	.75000
4	5/6	.83333
4	11/12	.91666
Between nodes of degree 4		
2	5/8	.62500
4	3/4	.75000
Between nodes of degree 3 and 4		
1	13/24	.54167
3	19/24	.79167

Table C.8: Rhombic dodecahedron

Distance	Resistance (Exact)	Resistance (Decimal)
Between nodes of degree 3		
2	67/90	0.74444
2	151/180	0.83889
4	43/45	0.95556
4	179/180	0.99444
6	61/60	1.01667
Between nodes of degree 5		
2	11/20	0.55000
4	7/10	0.70000
6	3/4	0.75000
Between nodes of degree 3 and 5		
1	31/60	0.51667
3	3/4	0.75000
3	4/5	0.80000
5	13/15	0.86667

Table C.9: Rhombic triacontahedron